RECEIVED

AUG 4 2000

OFFICE OF PETITIONS

Please type a plus sign (+) inside this box ⇒ +

PTO/SB/05 (1/98)
Approved for use through 09/30/00 OM 0651-0032
Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

UTILITY PATENT APPLICATION TRANSMITTAL (Only for new nonprovisional applications under 37 CFR 1 53(b))		
	Attorney Docket No.	**SP-1391-1D US**
	First Named Inventor or Application Identifier	Marc Tremblay
	Title	"A Grouping Logic Circuit In A Pipelined Superscalar Processor"
	Express Mail Label No.	**EL 487 695 809 US**

09/583097 08/02/99

APPLICATION ELEMENTS
See MPEP chapter 600 concerning utility patent application contents

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
Washington, D.C. 20231

1. ☒ Fee Transmittal Form - ***see page 2 of this form.***
   (Submit an original, and a duplicate for fee processing)
2. Application:
   ☒ Specification: (preferred arrangement set forth below)
   Descriptive title of the Invention,
   Cross References to Related Applications,
   Reference to Microfiche Appendix,
   Background of the Invention,
   Brief Summary of the Invention,
   Brief Description of the Drawings, and
   Detailed Description (all totaling 12 pages)
   Appendix(ces) ___, ___, & ___ (___pages)
   ☒ Claim(s) 2 pages
   ☒ Abstract of the Disclosure 1 page
3. ☒ Drawing(s) (35 USC 113) [Total Sheets 2]
4. Oath or Declaration ☐ unsigned [Total Pages ___]
   a. ☐ Newly executed (original or copy)
   b. ☒ Copy from prior application (37 CFR §1.63(d))
   (for continuation/divisional with Box 17 completed)
   c. ☐ DELETION OF INVENTOR(S)
   Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1 63(d)(2) and 1 33(b).
5. ☒ Incorporation By Reference (useable if Box 4b is checked)
   The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.
6. ☐ Microfiche Computer Program Appendix consisting of ____ pages of microfiche containing ___ frames on each page in accompanying envelope.
7. Nucleotide and/or Amino Acid Sequence Submission
   (if applicable, all necessary)
   a. ☐ Computer Readable Copy
   b. ☐ Paper Copy (identical to computer copy)
   c. ☐ Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

8. ☒ Assignment Papers (cover sheet & documents) 2 pages
9. ☐ 37 CFR §3.73(b) Statement (combined when there is an Assignee) ☒ Power of Attorney with Patent Declaration above.)
10. ☐ English Translation Document (if applicable)
11. ☐ Information Disclosure Statement (IDS) & ☐ PTO-1449 ☐ ___Copies of IDS Citations/References
12. ☒ Preliminary Amendment 2 pages
13. ☒ Return Receipt Postcard (MPEP 503) (should be specifically itemized)
14. Small Entity Status
    ☐ Small Entity Statement Enclosed ___ pages
    ☐ Statement filed in prior application; and status still proper and desired
    ☐ Is no longer claimed.
15. ☐ Certified Copy of Priority Document(s) (if foreign priority is claimed)
16. ☐ Other:
    ☐ Copy of Petition for Extension of Time filed in parent appln.;
    ☒ Petition Under 37 C.F.R. §1.53(e) (1 page).

17. If a CONTINUING APPLICATION, check appropriate box and supply the requisite information and a preliminary amendment:
☐ Continuation ☒ Divisional of prior application No. 08/662,582
Filed on June 11, 1996, entitled: "A Grouping Logic Circuit In A Pipelined Superscalar Processor".
PRIOR APPLICATION INFORMATION: *Examiner:* **Patel, G.** *Group Art Unit:* **2783**

18. CORRESPONDENCE ADDRESS

☐ Customer Number or Bar Code Label or ☒ Correspondence address below

Name	**Edward C. Kwok**			Reg. No. 33,938	
Attorneys for Applicant	**Skjerven Morrill MacPherson LLP**				
Address	25 Metro Drive, Suite 700				
City	San Jose	State	CA	Zip Code	95110
Country:	United States	Telephone	(408) 453-9200	Fax	(408) 453-7979

08/662,582

09583097 . 080200

Please type a plus sign (+) inside this box ⇒ [+]

PTO/SB/29 (1/98)
Approved for use through 09/30/00. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

19. Fee calculations.

CLAIMS (Number Filed)	(1) FOR	(2)		(3) NUMBER EXTRA		(4) RATE		(5) CALCULATIONS
9	TOTAL CLAIMS (37 CFR 1.16(c))	-20	=	0	X	$18	=	$ 0.00
1	INDEPENDENT CLAIMS (37 CFR 1.16(b))	-3	=	0	X	$78	=	$ 0.00
☐	MULTIPLE DEPENDENT CLAIMS (if applicable) (37 CFR 1.18(d))				+	$260.00	=	
		BASIC FEE (37 CFR 1.16(a))					=	$ 690.00
		Total of above Calculations					=	$ 690.00
	Reduction by 50% for filing by small entity (Note 31 CFR 1.9, 1.27, 1.28).						=	
	TOTAL						=	$ 690.00

20. FEES: The Commissioner is hereby authorized to credit overpayments or charge the following fees to Deposit Account No. **19-2386**:
   a. ☒ Fees required under 37 CFR 1.16. (U.S. Application Filing Fees)
   b. ☒ Fees required under 37 CFR 1.17. (Conditional Extension of Time Fees)
   c. ☐ Fees required under 37 CFR 1.18. (Patent Issue Fees)

21. ☒ Other: Petition Under 37 C.F.R. §1.53(e) $130.00

NOTE: **The prior application's correspondence address will carry over to this UPA UNLESS a new correspondence address is provided below.**

22. NEW CORRESPONDENCE ADDRESS

☐ Customer Number or Bar Code Label ☐ New correspondence address below

NAME					
ADDRESS					
CITY		STATE		ZIP CODE	
COUNTRY		TELEPHONE		FAX	

23. SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT REQUIRED

Skjerven Morrill MacPherson LLP
25 Metro Drive, Suite 700
San Jose, CA 95110
Tel. (408) 453-9200 Fax. (408) 453-7979

Date:	July 25, 2000	
Name	Edward C. Kwok	Reg. No. 33,938
Signature		
Express Mail Label No.	**EL 487 695 809 US**	

655230 v1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Tremblay, Marc

Assignee: Sun Microsystems, Inc.

Title: A Grouping Logic Circuit In A Pipelined Superscalar Processor

Serial No.: Unassigned   Filing Date: Herewith

Examiner: Unknown   Group Art Unit: Unknown

Docket No.: SP-1391-1D US

San Jose, California
July 25, 2000

BOX DAC
COMMISSIONER FOR PATENTS
Washington, D. C. 20231

PRELIMINARY AMENDMENT

Dear Sir:

Please amend the above-referenced patent application as follows:

IN THE CLAIMS

Please amend Claim 1 as follows:

1. (Amendment) A central processing unit, comprising:

a plurality of functional units, each functional unit adapted to execute an instruction of said central processing unit; and

a grouping logic circuit, including a number of pipeline stages and receiving, at each processor cycle, a group of instructions and one or more state vectors each representing states of instructions previously received at said grouping logic circuit in a preceding processor cycle wherein, based on said state vectors, said grouping logic circuit [dispatching] dispatches each of said currently received instructions to be

LAW OFFICES OF
SKJERVEN MORRILL
MACPHERSON LLP

25 METRO DRIVE
SUITE 700
SAN JOSE, CA 95110
(408) 453-9200
FAX (408) 453-7979

executed by one of said functional units, and provides a current state vector representing states of instructions of said currently received instructions.

REMARKS

Claims 1-9 are pending. Claim 1 is amended to more particularly point out and distinctly claim Applicant's invention.

If the Examiner has any question regarding the above, the Examiner is respectfully requested to telephone the undersigned Attorney for Applicants at 408-453-9200.

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail, Express Mail Label No. EL 487 695 809 US addressed to: Commissioner for Patents, Box DAC, Washington, D.C. 20231, on **July 25, 2000**.

7/25/2000

Attorney for Applicant | Date of Signature

Respectfully submitted,

Edward C. Kwok
Attorney for Applicant
Reg. No. 33,938

LAW OFFICES OF
SKJERVEN MORRILL
MACPHERSON LLP

25 METRO DRIVE
SUITE 700
SAN JOSE, CA 95110
(408) 453-9200
FAX (408) 453-7979

PIPELINED INSTRUCTION DISPATCH UNIT IN A SUPERSCALAR PROCESSOR

RECEIVED
AUG 4 2000
OFFICE OF PETITIONS

Marc Tremblay

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to computer architecture. In particular, this invention relates to the design of an instruction unit in a superscalar processor.

2. Discussion of the Related Art

Parallelism is extensively exploited in modern computer designs. Among these designs are two distinct architectures which are known respectively as the very long instruction word (VLIW) architecture and the superscalar architecture. A superscalar processor is a computer which can dispatch one, two or more instructions simultaneously. Such a processor typically includes multiple functional units which can independently execute the dispatched instructions. In such a processor, a control logic circuit, which has come to be known as the "grouping logic" circuit, determines the instructions to dispatch (the "instruction group"), according to certain resource allocation and data dependency constraints. The task of the computer designer is to provide a grouping logic circuit which can dynamically evaluate such constraints to dispatch instruction groups which optimally use the available resources. A resource allocation constraint can be, for instance, in a computer with a single floating point multiplier unit, the constraint that no more than one floating point multiply instruction is to be dispatched for any given processor cycle. A processor cycle is the basic timing unit for a pipelined unit of the processor, typically the clock period of the CPU clock. An example of a data

dependency constraint is the avoidance of a "read-after-write" hazard. This constraint prevents dispatching an instruction which requires an operand from a register which is the destination of an write instruction dispatched earlier, but yet to be unretired.

A VLIW processor, unlike a superscalar processor, does not dynamically allocate system resources at run time. Rather, resource allocation and data dependency analysis are performed during program compilation. A VLIW processor decodes the long instruction word to provide the control information for operating the various independent functional units. The task of the compiler is to optimize performance of a program by generating a sequence of such instructions which, when decoded, efficiently exploit the program's inherent parallelism in the computer's parallel hardware. The hardware is given little control of instruction sequencing and dispatch.

A VLIW computer, however, has a significant drawback in that its programs must be recompiled for each machine they run on. Such recompilation is required because the control information required by each machine is encoded in the instruction words. A superscalar computer, by contrast, is often designed to be able to run existing executable programs (i.e., "binaries"). In a superscalar computer, the instructions of an existing executable program are dispatched by the computer at run time according to the computer's particular resource availability and data integrity requirements. From a computer user's point of view, because existing binaries represent significant investments, the ability to acquire enhanced performance without the expense of purchasing new copies of binaries is a significant advantage.

In the prior art, to determine the instructions that go into an instruction group of a given processor

cycle, a superscalar computer performs the resource allocation and data dependency checking tasks in the immediately preceding processor cycle. Under this scheme, the computer designer must ensure that such resource allocation and data dependency checking tasks complete within their processor cycle. As the number of the functional units that can be independently run increases, the time required for performing such resource allocation and data dependency checking tasks grows more rapidly than linearly. Consequently, in a superscalar computer design, the ability to perform resource and data integrity analysis within a single processor cycle can become a factor that limits the performance gain of additional parallelism.

SUMMARY OF THE INVENTION

The present invention provides a central processing unit which includes a grouping logic circuit for determining simultaneously dispatchable instructions in an processor cycle. The central processing unit of the present invention includes such a grouping logic circuit and a number of functional units, each adapted to execute one or more specified instructions dispatched by the grouping logic circuit. The grouping logic circuit includes a number of pipeline stages, such that resource allocation and data dependency checks can be performed over a number of processor cycles. The present invention therefore allows dispatching a large number of instruction simultaneously, while avoiding the complexity of the grouping logic circuit from becoming limiting the duration of the central processing unit's processor cycle.

In one embodiment, the grouping logic circuit checks intra-group data dependency immediately upon receiving the instruction group. In that embodiment, all instruction in a group of instructions received in

a first processor cycle are dispatched prior to dispatching any instruction of a second group of instructions received at an processor cycle subsequent to said first processor cycle.

The present invention is better understood upon consideration of the detailed description below in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a CPU 100, in an exemplary 4-way superscalar processor of the present invention.

Figure 2 shows schematically a 4-stage pipelined grouping logic circuit 109 in the 4-way superscalar processor of Figure 1.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

An embodiment of the present invention is illustrated by the block diagram of Figure 1, which shows a central processing unit (CPU) 100 in an exemplary 4-way superscalar processor of the present invention. A 4-way superscalar processor fetches, dispatches, executes and retires up to four instructions per processor cycle. As shown in Figure 1, central processing unit 100 includes two arithmetic logic units 101 and 102, a load/store unit 103, which includes a 9-deep load buffer 104 and an 8-deep store buffer 105, a floating point adder 106, a floating point multiplier 107, and a floating point divider 108. In this embodiment, a grouping logic circuit 109 dispatches up to four instructions per processor cycle. Completion unit 110 retires instructions upon completion. A register file (not shown), including numerous integer and float point registers, is provided with sufficient number of ports to prevent contention among functional units for access to this register file during operand fetch or result write-back. In this

embodiment also, loads are non-blocking, i.e., CPU 100 continues to execute even though one or more dispatched load instructions have not complete. When the data of the load instructions are returned from the main memory, these data can be placed in a pipeline for storage in a second-level cache. In this embodiment, floating point adder 106 and floating point multiplier 107 each have a 4-stage pipeline. Similarly, load/store unit 103 has a 2-stage pipeline. Floating point divider 108, which is not pipelined, requires more than one processor cycle per instruction.

To simplify the discussion below, the state of CPU 100 relevant to grouping logic 109 is summarized by a state variable S(t), which is defined below. Of course, the state of CPU 100 includes also other variables, such as those conventionally included in the processor status word. Those skilled in the art would appreciate the use and implementation of processor states. Thus, the state S(t) at time t of CPU 100 can be represented by:

$$S(t) = \{ALU_1(t), ALU_2(t), LS(t), LB(t), SB(t), FA(t), FM(t), FSD(t)\}$$

where $ALU_1(t)$ and $ALU_2(t)$ are the states, at time t, of arithmetic logic units 101 and 102 respectively; LS(t) and LB(t) are the states, at time t, of store buffer 105 and load buffer 104 respectively; FA(t), FM(t), and FDS(t) are the states, at time t, of floating point adder 106, floating point multiplier 107 and floating point divider 108 respectively.

At any given time, the state of each functional unit can be represented by the source and destination registers specified in the instructions dispatched to the functional unit but not yet retired. Thus,

$$ALU_1 = \{ALU_1.rs1(t),\ ALU_1.rs2(t),\ ALU_1.rd(t)\}$$

where rs1(t), rs2(t) and rd(t) are respectively the first and second source registers, and the destination of registers of the instruction executing at time t in arithmetic logic unit 101.

Similarly, the state of arithmetic logic unit 102 can be defined as:

$$ALU_2 = \{ALU_2.rs1(t),\ ALU_2.rs2(t),\ ALU_2.rd(t)\}$$

For pipelined functional units, such as floating point adder 106, the state is relatively more complex, consisting of the source and destination registers of the instructions in their respectively pipeline. Thus, for the pipelined units, i.e., load/store unit 103, load buffer 104, store buffer 105, floating point adder 106, and floating point multiplier 107, their respective states, at time t, LS(t), LB(t), SB(t), FA(t) and FM(t) can be represented by:

$$LS = \{LS.rs1_i(t),\ LS.rs2_i(t),\ LS.rd_i(t)\} \quad for\ i=\{1,\ 2\}$$

$$LB = \{LB.rs1_i(t),\ LB.rs2_i(t),\ LB.rd_i(t)\} \quad for\ i=\{1,2,\ldots,9\}$$

$$SB = \{SB.rs1_i(t),\ SB.rs2_i(t),\ SB.rd_i(t)\} \quad for\ i=\{1,2,\ldots,8\}$$

$$FA = \{FA.rs1_i(t),\ FA.rs2_i(t),\ FA.rd_i(t)\} \quad for\ i=\{1,\ldots,4\}$$

Finally, floating point divider 108's state FSD(t)

$$FM = \{FM.rs1_i(t),\ FM.rs2_i(t),\ FM.rd_i(t)\} \quad for\ i=\{1,...,4\}$$

can be represented by:

$$FDS = \{FDS.rs1_i(t),\ FDS.rs2_i(t),\ FDS.rd_i(t)\}$$

State variable S(t) can be represented by a memory element, such as a register or a content addressable memory unit, at either a centralized location or in a distributed fashion. For example, in the distributed approach, the portion of state S(t) associated with a given functional unit can be implemented with the control logic of the functional unit.

In the prior art, a grouping logic circuit would determine from the current state, S(t) at time t, the next state S(t+1), which includes information necessary to dispatch the instructions of the next processor cycle at time t+1. For example, to avoid a read-after-write hazard, such a grouping circuit would exclude from the next state S(t+1) an instruction having an operand to be fetched from a register designated for storing a result of a yet incomplete instruction. As another example, such a grouping circuit would include in state S(t+1) no more than one floating point "add" instruction in each processor cycle, since only one floating point adder (i.e. floating point adder 106) is available. As discussed above, as complexity increases, the time required for propagating through the grouping logic circuit can become a critical path for the processor cycle. Thus, in accordance with the present invention, grouping logic circuit 109 is pipelined to derive, over τ processor cycles, a future state $S(t+\tau)$ based on the present state S(t). The future state $S(t+\tau)$ determines the instruction group to dispatch at time $t+\tau$. Pipelining grouping logic 109 is possible because, as demonstrated below, (i) the values of most state variables in the state $S(t+\tau)$ can be

estimated from corresponding values of state S(t) with sufficient accuracy, and (ii) for those state variables for which values can not be accurately predicted, it is relatively straightforward to provide for all possible outcomes of state $S(t+\tau)$, or to use a conservative approach (i.e. not dispatching an instruction when such an instruction could have been dispatched) with a slight penalty on performance.

The process for predicting state $S(t+\tau)$ is explained next. The following discussion will first show that most components of next state S(t+1) can be precisely determined from present state S(t), and the remaining components of state S(t) can be reasonably determined, provided that certain non-deterministic conditions are appropriately handled. By induction, it can therefore be shown that future state $S(t+\tau)$, where τ is greater than 1, can likewise be determined from state S(t).

Since an instruction in floating point adder 106 or floating point multiplier 107 completes after four processor cycles and an instruction in load/store unit 103 completes after two processor cycles, the states FA, FM and LS at time t+1 can be derived from the corresponding state S(t) at time t, the immediately preceding processor cycle. In particular, the relationship governing the source and destination registers of each instruction executing in floating point adder 106, floating point multiplier 107 and load/store unit 103 between time t+1 and time t are:

$$rs1_i(t+1) = rs1_{i-1}(t), \quad for\ 1<i\leq k$$

$$rs2_i(t+1) = rs2_{i-1}(t), \quad for\ 1<i\leq k$$

$$rd_i(t+1) = rd_{i-1}(t), \quad for\ 1<i\le k$$

where k is the depth of the respective pipeline.

The state FSD(t+1) of floating point divider 108, in which the time required to execute an instruction can exceed an processor cycle, is determined from state FSD(t) by:

$$FSD(t+1) = FSD(t)\{\ if\ last\ stage\}\ else\ null$$

Whether or not floating point divider 108 is in its last stage can be determined from, for example, a hardware counter or a state register, which keep tracks of the number of processor cycles elapsed since the instruction in floating point divider 108 began execution.

In load buffer 104 and store buffer 105, since the pending read or write operation at the head of each queue need not complete within one processor cycle, the state LB(t+1) at time t+1 cannot be determined from the immediately previous state LB(t) at time t with certainty. However, since state LB(t+1) can only either remain the same, or reflect the movement of the pipeline by one stage, two possible approaches to determine state LB(t+1) can be used. First, a conservative approach would predict LB(t+1) to be the same as LB(t). Under this approach, when load buffer 104 is full, an instruction is not dispatched until the pipeline in load buffer 106 advances. An incorrect prediction, i.e. a load instruction completes during the processor cycle of time t, this conservative approach leads to a penalty of one processor cycle, since a load instruction could have been dispatched at time t+1. Alternatively, a more aggressive approach provides for both outcomes, i.e. load buffer 104 advances one stage, and load buffer 104 remains the

same. Under this aggressive approach, grouping logic 109 is ready to dispatch a load instruction, such dispatch to be enabled by a control signal which indicates, at time t+1, whether a load instruction has in fact completed. This aggressive approach requires more a complex logic circuit than the conservative approach.

Thus, the skilled person would appreciate that state S(t+1) of CPU 100 can be predicted from state S(t). Consequently, both the number of instructions and the types of instructions that can be dispatched at time t+1 (i.e. the instruction group at time t+1) based on predicted state S(t+1) can be derived, at time t, from state S(t), subject to additional handling based on the actual state S_A(t+1) at time t+1.

The above analysis can be can be extended to allow state S(t+τ) at time t+τ to be derived from state S(t) at time t. The instruction group at time t+τ can be derived from time t, provided that, for each instruction group between time t and t+τ, all instruction from that instruction group must be dispatched before any instruction from a subsequent instruction group is allowed to be dispatched (i.e. no instruction group merging).

Since instructions from different instruction groups are not merged, intra-group dependencies and inter-group dependencies can be checked in parallel. The instructions are either fetched from an instruction cache or an instruction buffer. An instruction buffer is preferable in a system in which not all accesses (e.g. branch instructions) to the instruction cache are aligned, and multiple entry points in the basic blocks of a program are allowed.

Once four candidate instructions for an instruction group are identified, intra-group data dependency checking can begin. Because of the constraint against instruction group merging described

above, i.e., all instructions in an instruction group must be dispatched before an instruction from a subsequent instruction group can be dispatched, intra-group dependency checking can be accomplished in a pipelined fashion. That is, intra-group dependency checking can span more than one processor cycle and all inter-group dependency checking can occur independently of inter-group dependency checking. For the purpose of intra-group dependency check, each instruction group can be represented by:

$$IntraS(t) = \{rs1_i(t),\ rs2_i(t),\ rd_i(t),\ res_i(t)\} \quad \text{for } 0 \le i < W-1$$

where W is the width of the machine, and res_i represents the resource utilization of instruction I. An example of a four-stage pipeline 200 is shown in Figure 2. In Figure 2, at first stage 201, as soon as the instruction group is constituted, intra-group dependency checking is performed immediately. Thereafter, at stage 202, resource allocation within the instruction group can be determined. At stage 203, intergroup decisions, e.g. resource allocation decisions taking into consideration resource allocation in previous instruction groups, are merged with the decisions at stages 201 and 202. For example, if the present instruction group includes an instruction designated for floating point divider 108, stage 203 would have determined at by this time if a previous instruction using floating point divider 108 would have completed by the time the present instruction group is due to be dispatched. Finally, at stage 204, non-deterministic conditions, e.g. the condition at store buffer 105, is considered. Dispatchable instructions are issued into CPU 100 at the end of stage 204.

The above detailed description is provided to

illustrate the specific embodiments of the present invention and is not intended to be limiting. Numerous variations and modifications within the scope of the present invention are possible. The present invention is defined by the following claims.

CLAIMS

I claim:

1. A central processing unit, comprising:

a plurality of functional units, each functional unit adapted to execute an instruction; and

a grouping logic circuit, including a number of pipeline stages and receiving, at each processor cycle, a group of instructions, said grouping logic circuit dispatching each of said instructions to be executed by one of said functional units.

2. A central processing unit as in Claim 1, wherein said grouping logic circuit checks data dependency among said group of instructions to determine whether said group of instructions can be dispatched simultaneously.

3. A central processing unit as in Claim 1, wherein said grouping logic circuit checks for resource contention within said group of instructions.

4. A central processing unit as in Claim 1, wherein said grouping logic circuit checks data dependency of an instruction group at one processor cycle and a group of instruction received in a previous processor cycle.

5. A central processing unit as in Claim 1, wherein the state of said central processing unit is represented in a register, said state including representation of destination registers of instructions in said group of instructions.

6. A central processing unit as in Claim 1, wherein all instruction in a group of instructions

received in a first processor cycle are dispatched prior to dispatching any instruction of a second group of instructions received at an processor cycle subsequent to said first processor cycle.

7. A central processing unit as in Claim 1, wherein said functional units include a pipelined functional unit capable of receiving an instruction every processor cycle and completing said instruction at a subsequent processor cycle.

8. A central processing unit as in Claim 1, wherein said functional units include a functional unit requiring multiple processor cycles to complete an instruction executed at said functional unit.

9. A central processing unit as in Claim 1, wherein said grouping logic circuit derives a state vector for a group of instructions received at a first processor cycle based on a number of state vectors derived for groups of instructions received in a number of processor cycles immediately preceding said first processor cycle, said number of processor cycles being equal to said number of pipeline stages.

PIPELINED INSTRUCTION DISPATCH UNIT
IN A SUPERSCALAR PROCESSOR

Marc Tremblay

ABSTRACT OF THE DISCLOSURE

A pipelined instruction dispatch or grouping circuit allows instruction dispatch decisions to be made over multiple processor cycles. In one embodiment, the grouping circuit performs resource allocation and data dependency checks on an instruction group, based on a state vector which includes representation of source and destination registers of instructions within said instruction group and corresponding state vectors for instruction groups of a number of preceding processor cycles.

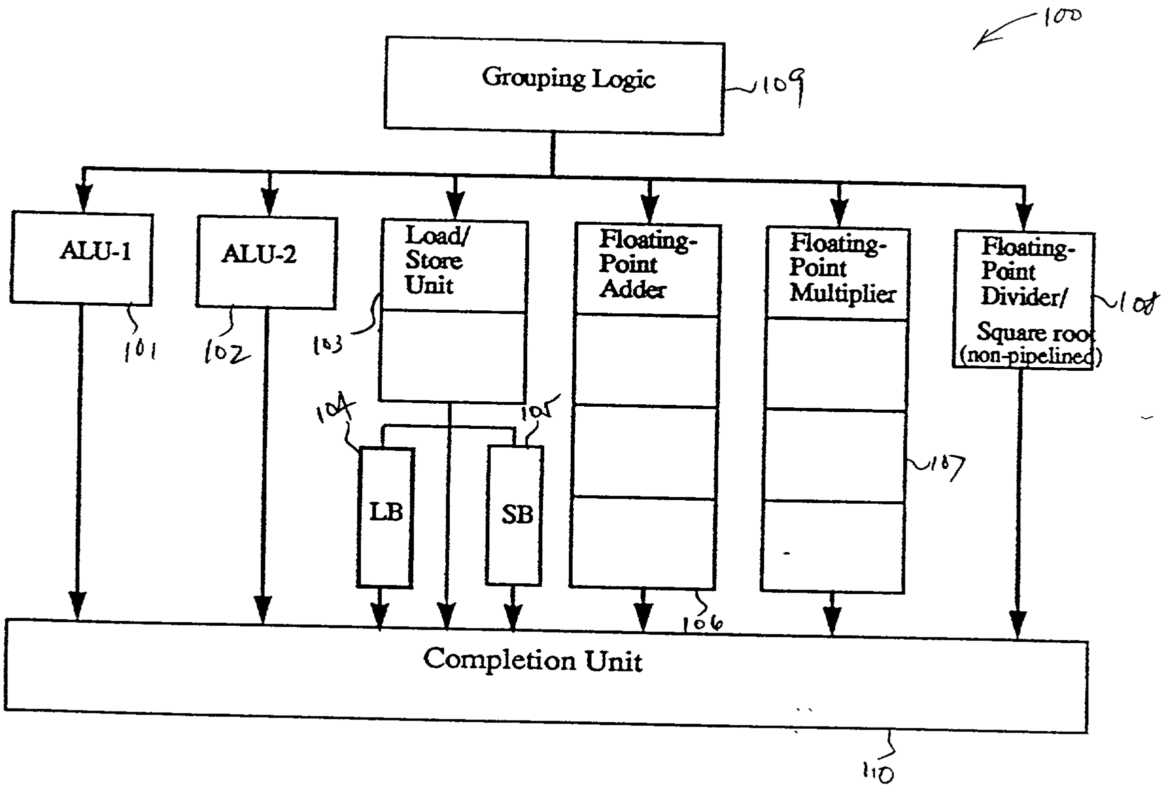
100
Grouping Logic
109
ALU-1
ALU-2
Load/
Store
Unit
Floating-
Point
Adder
Floating-
Point
Multiplier
Floating-
Point
Divider/
Square root
(non-pipelined)
108
101
102
103
104
105
LB
SB
107
106
Completion Unit
110

FIGURE 1

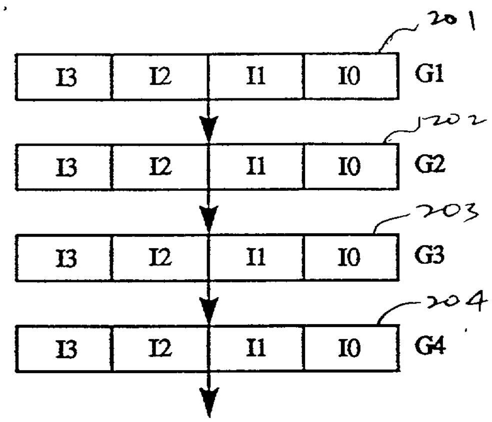
201
I3
I2
I1
I0
G1
202
I3
I2
I1
I0
G2
203
I3
I2
I1
I0
G3
204
I3
I2
I1
I0
G4

FIGURE 2

Atty. Docket No. M-3876 US

DECLARATION FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below adjacent to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of subject matter (process, machine, manufacture, or composition of matter, or an improvement thereof) which is claimed and for which a patent is sought by way of the application entitled PIPELINED INSTRUCTION DISPATCH UNIT IN A SUPERSCALAR PROCESSOR

which (check) [X] is attached hereto.
[ ] and is amended by the Preliminary Amendment attached hereto.
[ ] was filed on ______ as Application Serial No. ______
[ ] and was amended on ______ (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information known to me to be material to the examination of this application in accordance with Title 37, Code of Federal Regulations, §1.56(a).

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s) Priority Claimed

(Number)	(Country)	(Day/Month/Year Filed)	Priority Claimed
N/A			Yes No
			Yes No
			Yes No

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as any subject matter of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §1.56(a) which occurred between the filing date of the prior application(s) and the national or PCT international filing date of this application:

(Application Serial No.)	(Filing Date)	(Status-patented, pending, abandoned)
N/A		

Rev. 950707

AUG 15 2000

Atty. Docket No. M-3876 US

I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and to transact all business in the United States Patent and Trademark Office connected therewith:

Alan H. MacPherson (24,423); Thomas S. MacDonald (17,774); Richard Franklin (19,128); Kenneth E. Leeds (30,566); Paul J. Winters (25,246); Brian D. Ogonowsky (31,988); David W. Heid (25,875); Guy W. Shoup (26,805); Forrest E. Gunnison (32,899); Norman R. Klivans (33,003); Edward C. Kwok (33,938); Patrick T. Bever (33,834); David E. Steuber (25,557); Michael Shenker (34,250); Laura Terlizzi (31,307); T. Lester Wallace (34,748); Ronald J. Meetin (29,089); Andrew C. Graham (36,531); Ken John Koestner (33,004); Stephen A. Terrile (32,946); Omkar K. Suryadevara (36,320); David T. Millers (37,396); E. Eric Hoffman (38,186); Kent B. Chambers (38,839); Emily M. Haliday (38,903); William L. Paradice, III (38,990); Arthur J. Behiel (39,603); Serge J. Hodgson (40,017); David W. O'Brien (40,107); Lee Patch (30,095); Matthew C. Rainey (32,291); James W. Rose (34,239); Erwin J. Basinski (34,773); Kang S. Lim (37,491); Timothy J. Crean (37,116); Leland Z. Wiesner (39,424); Philip J. McKay (38,966).

Address all correspondence and telephone calls to:

Edward C. Kwok
Attorney for Applicant(s)
SKJERVEN, MORRILL, MacPHERSON, FRANKLIN & FRIEL
25 Metro Drive, Suite 700
San Jose, California 95110-1349

Telephone: 408-453-9200
Facsimile: 408-453-7979

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Title 18, United States Code, § 1001 and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor Marc Tremblay
Inventor's signature [signature] Date 6/6/96
Residence Palo Alto, California Citizenship Canada
Post Office Address 801 Waverley Street, #3
Palo Alto, California 94301

E:\DMS\8242\M-3876_U\0172397.WP

FAX RECEIVED
AUG 15 2000